N90-22316

KNOWLEDGE STRUCTURE REPRESENTATION AND AUTOMATED UPDATES IN INTELLIGENT INFORMATION MANAGEMENT SYSTEMS

Stephen M. Corey
Richard S. Carnahan, Jr.

Martin Marietta Information & Communications Systems
Denver, Colorado

1.0 ABSTRACT

Work reported in this paper is part of a continuing effort to apply rapid prototyping and Artificial Intelligence techniques to problems associated with projected Space Station-era information management systems. In particular, timely updating of the various databases and knowledge structures within our proposed intelligent information management system (IIMS) is critical to support decision making processes. Because of the significantly large amounts of data entering the IIMS on a daily basis, information updates will need to be automatically performed with some systems requiring that data be incorporated and made available to users within a few hours. Meeting these demands depends first, on the design and implementation of information structures that are easily modified and expanded, and second, on the incorporation of intelligent automated update techniques that will allow meaningful information relationships to be established. This paper examines potential techniques for developing such an automated update capability and examines IIMS update requirements in light of results obtained from our IIMS prototyping effort.

2.0 INTRODUCTION

The advent of large, information intensive data systems will require sophisticated user access capabilities. In particular, projected large data volumes necessitate implementation of extensive browsing and querying facilities. Indeed, given the prospect of multiterabyte databases (which will be continually updated), it is our contention that without the availability of higher-level information to help focus a user's search, data access would be virtually impossible. To provide rapid access in a data environment whose structure can be transparently and dynamically altered, we employ the conceptual structure of *metadata*, or information about the data, as the main information structure. Additionally, to allow for maximum efficiency in the incorporation of new information, *metadata* structures must be automatically generated and maintained. When an update message is received indicating the arrival of new data, the message is processed inferring new *metadata* that is then appropriately linked to already existing *metadata* in the knowledge structure. As we have mentioned before (Carnahan, Corey, & Snow, 1989), the speed at which this process can be accomplished depends on the type of data received and its potential relationships to other *metadata*. Following sections describe the approaches we have taken to better define requirements for an advanced information management *metadata* knowledge structure and automated update system.

3.0 TYPICAL QUERY FORMULATION

In typical query systems, much is required of the user. Before users can formulate queries to locate data sets related to their areas of interest, a mathematically oriented query language must generally be learned. In addition, users must become familiar with the physical or logical view of the data structure. In a relational database management system (DBMS), this process requires learning attribute and relation names, as well as relationships between them. While such a condition is acceptable for small systems containing few relations and attributes, the situation rapidly becomes unmanageable for very large database systems.

Recently, natural language (NL) query formulation systems have been implemented to aid in the elimination of the need for a mathematical query system. Nevertheless, even though the query language associated with NL systems is not mathematical, the user is still required to learn a query language of sorts. As a result, words and phrases recognized by the NL system grammar and appropriate procedures for combining them must be learned. Unfortunately, users must face other problems when using a NL system. NL systems must be trained for a specific domain under which it will operate. Domain terminology is taught to the grammar so that a specific user may converse in a manner common to that domain. Terminology, however, is often dissimilar among scientific domains; NL systems are generally not well suited for multidisciplinary information. Yet it must be remembered that even if problems associated with different domain terminology are resolved, the user is still required to learn the underlying data structure and its relationships.

With either the NL or the typical query system, the user forms queries without complete knowledge of data contained in the database. In this type of an environment, results of the query may contain data sets of no interest, or the results may contain only some relevant data sets. The former consequence results from

either inadequate or inappropriate types of constraints being placed on the query. Hence, the querying process is not completed and the user has to sift through inappropriate data to locate data sets of interest. The latter consequence is more serious; some of the data sets of interest cannot be found because of constraints placed on the query by a user who does not fully comprehend the parameters of the targeted data sets. Furthermore, a null result is also possible and the user is left to wonder at the cause for lack of system response. Database research applied to this issue has resulted in prototype systems in which the user is provided with an understanding of which subclause(s) of the query has caused the inappropriate elimination of data sets, but commercially available DBMSs have not yet implemented this feature. Yet, even when such features do become available, information provided concerning the failed query is *post hoc*; no real aid is provided to the user while the query is being formed. For users who are unfamiliar with the data, the task of forming an appropriate query will be formidable. In light of problems associated with current systems, we have taken a different approach.

4.0 IIMS QUERY FORMULATION

To address the limitations of typical query systems and notably, those encountered when accessing extremely large databases, the query system of the prototype Intelligent Information Management System (IIMS) has been implemented using a *metadata* base (rather than a database containing actual data), and with a querying approach called *assisted query formulation*.

Since databases in which data sets are to be located will likely be extremely large, geographically distributed, and heterogeneous, the possibility of providing a real-time interface with standard techniques is, at best, remote. One way to address this problem is to employ a method that reduces the amount of information to be searched. A typical information reduction technique used in standard database systems is the use of indices. We believe, however, that the exclusive use of this technique will not adequately reduce access time in the type of query environment we envision. In contrast, the IIMS achieves information reduction through the appropriate selection of a small abstraction of all possible information contained in the databases; this abstraction of data we refer to as the *metadata* base. Through the use of a *metadata* base, not only is the amount of information to be accessed reduced, but relationship information among the data can also be included; a capability that would be impossible without abstraction due to overhead associated with this information. Thus, the *metadata* base represents not simply an abstraction of information but a knowledge base of *metadata* and *metadata* relationships, and provides the user with more extensive help to eliminate irrelevant information and increases the probability that all data sets of interest are located.

While the selection of a *metadata* base as the focus of access only serves to bring the problem of very large database access to a manageable level, it does nothing to address query problems encountered in normal systems. It seems apparent that a query system must be created that takes advantage of knowledge contained in the *metadata* base. With *assisted query formulation*, the user is guided through the process of formulating queries on the *metadata* base through menus that present only relevant information. Items displayed to the user are controlled by the underlying knowledge structure and are determined largely by the user's navigation path through the *metadata*. The user is never presented with selections that are not contained in the *metadata* base. Since query formulation is based entirely on the user's navigation path and selections, only valid queries can be formed.

This approach to navigation and query formulation frees the user from having to learn a query language since the knowledge structure provides the user with appropriate next selections that are syntactically and semantically valid. In addition, because the query system presents the user with value information contained within the database, the user never has to guess what an appropriate value is for attributes being presented. For example, when the user chooses the "Programs" *metadata* concept, valid values for this node in the knowledge structure (data collection programs) are presented as selections. The user does not have to be concerned about the form of the query input, whether it is a string, if it contains spaces or underscores, and the like. All valid potential query inputs are presented to the user in a menu that allows the desired item to be selected. Since such information is presented to the user as the query is being formed, the user is actually browsing information contained in the *metadata* base and forming the query based on actual information; as a result, a more accurate query is formed. This process reduces the number of probing queries users have to form before information for which they are searching is located. Furthermore, response time for each step in the query formulation process is rapid due to the fact that only a small amount of *metadata* is being processed at a given time.

Obviously, since projected databases are likely to be very large, not all value information can be contained in the knowledge structure; if so, the size of the *metadata* base would approach or exceed the size of the original databases. However, it is important for the *metadata* base to be able to contain more detailed information in some areas than in others. As a result, it is necessary for the knowledge structure to be able to handle multiple levels of *metadata* abstraction. During navigation, the user will be presented with value information only when available, and in a seamless, transparent manner.

Assisted query formulation also frees the user from having to understand the physical organization of information contained in the *metadata* base. Appropriate information is presented to the user when required and all the user must do is select from presented items. However, the user is not totally released from understanding information categorization. Before the user can select an item, the relative position of the item within the knowledge structure must be understood. Therefore, the user may know the concept (node) in the knowledge structure at which he desires to be positioned, but he may not be able to readily locate or navigate to the concept. To help resolve this problem, the IIMS prototype employs two features: first is the use of domain specific terminology and information organization; that is, information is presented to the user in a familiar manner and is achieved through the use of data views. Second, we have implemented a FIND capability that allows the user to select from a list of appropriate concepts. Only those concepts relevant to the user's interests are presented. Once the selection is made, the IIMS prototype locates the shortest path to the chosen concept from the user's current knowledge structure location and then automatically navigates to that concept. In the future, other navigation assistance tools addressing this issue will also be studied.

5.0 KNOWLEDGE-NET - THE IIMS KNOWLEDGE MODEL

To provide the types of capabilities described above and still satisfy the critical requirements of dynamic modification and update (which in most instances, we believe, will have to be performed automatically) we have implemented *metadata* knowledge and relationship information using a data representation rather than procedural approach. The data representation we are using can be categorized as a semantic network consisting of typed nodes and typed, unidirectional links. One unique feature of our implementation is that relationships have conditional relevance, that is, not all links emanating from a particular node are relevant under all conditions. Link relevance is computed dynamically and is based primarily on the navigation path traversed by the user through the knowledge structure. Determination of link relevance at any given time and its use in the query formulation process is discussed in section 5.5.

In the knowledge-net (k-net), *metadata* concepts or facts are represented as typed nodes while relationships between concepts are represented as typed links. Node or link characteristics and their effects on knowledge structure composition and use are governed by their types. In the current system, eight types of nodes (*Fact, Information, ISA-S, ISA-V, Paren, Operator, Structure, Value*) and ten types of links (*Abstraction, Acronym, Alias, Fact, Information, ISA, Reverse, Structure, Value, Value Selection*) are used. Of course, the number of types of both nodes and links is not fixed and will likely change as the k-net increases its expressive capabilities. Rather than discussing separately each of the typed nodes and links listed above, each will be discussed, whenever possible, in the context of its combined use with other nodes and links to represent knowledge. Primary knowledge representation capabilities of the k-net are implemented using *Fact, ISA-S, ISA-V, Structure* and *Value* nodes and *Abstraction, Fact, ISA, Structure, Value,* and *Value Selection* links. Other node and link types are generally used to define associated information, alternative names for domain concepts, and syntax representation (see section 4.2). Figure 1 illustrates a sample *value tree* from the current IIMS knowledge structure.

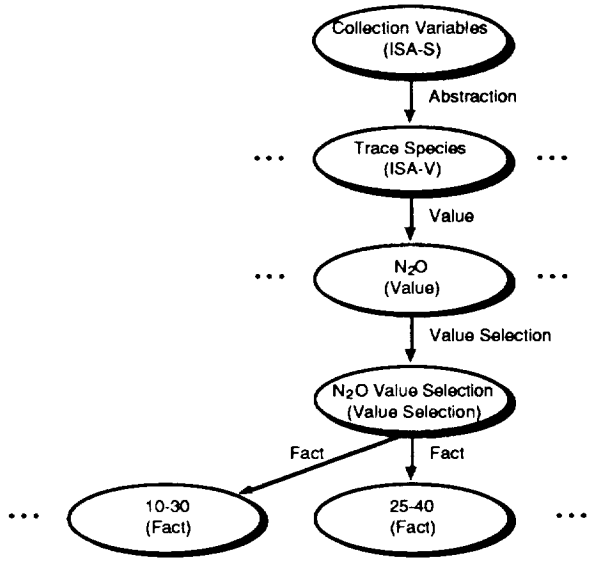

Figure 1 Example Knowledge Structure Value Tree

The current implementation of the k-net may be thought of as a series of *value trees* grouped together through the use of *Structure* nodes (while this is generally true, it should be noted that the knowledge structure is highly interconnected). Similar to any typical hierarchical representation of knowledge, *Structure* nodes are generally used as category headings. Similar to a directory structure, they are also used to represent a concept which does not directly have an associated value. Selection of structure nodes does not cause a change in the status of the query currently being formed and hence are not involved in any value processing performed.

5.1 VALUE TREES - ATTRIBUTE/VALUE BINDINGS

Within the k-net, *value trees* control the setting of attribute values (attributes are parameters describing appropriate data sets). Query formation is fundamentally the process of setting values or ranges of values for

known data set attributes and relationships among attributes. This process is achieved by navigating through relevant nodes of the k-net. However, no constraints are added to the query until entering a value tree and indicating the value for a specified attribute. Of course, this process is transparent to the user and navigation procedures are not perceptively different when a value tree is entered. *ISA-S* nodes are the attribute nodes of the k-net, and are the only nodes that can be assigned a value. Values of *ISA-S* nodes are designated by nodes below them in the tree and hence, they represent root nodes of small *value trees* embedded within the k-net. *Value trees* are generally never more than a few levels deep, but node values may be set at different levels within the *value tree*. This is caused by the iterative refinement of the original domain concept into more specific detail represented by the *value tree*. For example, in the current prototype k-net, the node "Collection Variables" is an *ISA-S* node which has "Trace Species" as one of the concept nodes below it (see Figure 1). The "Trace Species" node represents a valid value for the "Collection Variables" node whose value specification, at this point, can be terminated. However, if "Trace Species" is still too general, the user can select the "Trace Species" node and more specific concept nodes or values of "Trace Species" will be offered to the user. Upon selection of one of these more specific concept nodes, the value for the node "Collection Variables" is altered to reflect the new selection. A *Value* link is used to model a value relationship between two nodes, in which the destination node represents a legitimate value of the originator node. This relationship implies that the destination node represents an *ISA* type of the originator node. Because of this, whenever a *Value* link is defined, an *ISA* link is automatically defined in the opposite direction of the *Value* link. If the destination node of the *ISA* link (i.e., the originator node of the *Value* link) is a *Structure* or a *Value* node, the node type is altered to indicate that this node now represents an *ISA* concept. When a node type change is required, a *Structure* node is changed to an *ISA-S* node and a *Value* node is changed to an *ISA-V* node.

When the query formulation process arrives at the "Collection Variables" node, the root node of the example *value tree* illustrated in Figure 1, determining what concepts to present to the user is governed by the *ISA* concept structure mentioned above. Determining whether the concept "Trace Species" should be presented does not depend on the relevance of the link between "Collection Variables" and "Trace Species" but rather on the relevance of the links between "Trace Species" and "N_2O" and other related concepts. To represent this correctly, all RELEVANCE conditions (see section 5.5 for a discussion of RELEVANCE conditions) on links emanating from "Trace Species" would have to be associated with the link between "Collection Variables" and "Trace Species". Such a procedure is inefficient since multiple copies of the same relationship representation must be maintained. Instead of this approach, we have chosen to implement an *Abstraction* link, representing the notion that the relationship between nodes is an abstraction of the value of the node attribute being set.

Upon encountering an *Abstraction* link, the system progresses directly to the appropriate destination node and begins relevance processing on links emanating from that node. When one of these *Value* links is found to be relevant, given the current state of the k-net, the *Abstraction* link is then considered to be relevant. Abstraction relevance processing is an iterative process. If links from the destination node of the *Abstraction* link also contain *Abstraction* links, relevance processing then shifts to the destination node of the new *Abstraction* link. The process continues until no *Abstraction* links are encountered, at which point relevance processing then begins. Under appropriate conditions, *Abstraction* links, like the *ISA* link, are generated automatically whenever a *Value* link is defined. Figure 2 illustrates the state of the *value tree* at the time a request is received to define the *Value* link between "Trace Species" and "N_2O". Since the *Value* link between

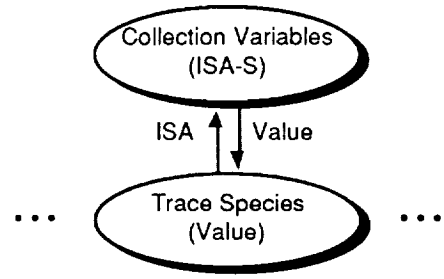

Figure 2 Value Tree State Prior to "Trace Species" Value Link Definition

"Collection Variables" and "Trace Species" represents the same relationship as the *Abstraction* link is intended to represent (see Figure 3), the *Value* link is deleted from the k-net and replaced with the *Abstraction* link. Thus, defining a *Value* link can generate three links, delete one link, and change a node type. Since relevance of the *Abstraction* link is determined by other links, *Abstraction* links are allowed only to contain view relevance and are processed using *view filtering* (see section 5.5).

Originally, *ISA-S* nodes are specified as *Structure* nodes but their type is changed to *ISA-S* when an *ISA* link creation request is received specifying the *Structure* node as the destination node of the link. Specification of the *ISA* link indicates that the *ISA-S* node represents a node value abstraction and, thus, has different characteristics than a typical *Structure* node.

When an *ISA* link is defined, an *Abstraction* link is automatically defined in the opposite direction. Determining the relevance (see section 5.5 for a discussion of relevance processing) of the *Abstraction* link should be based on the relevance of the links to the concept which is the destination of the *Abstraction* link. The *Abstraction* link is considered relevant if at least

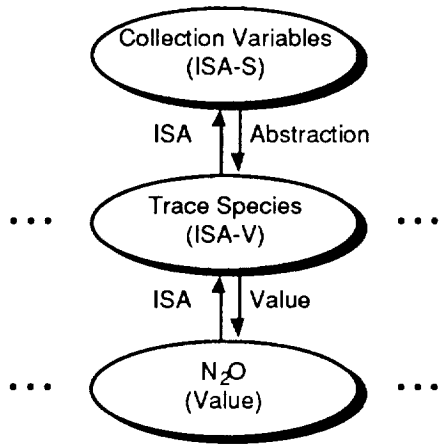

Figure 3 Value Tree State After "Trace Species" Value Link Definition

one of the value links of its destination node is relevant. *Abstraction* links are used to prevent maintaining duplicate copies of relevance instances and the situation where an *Abstraction* concept node is offered to the user, but upon its selection all further selections have been eliminated because of relevance processing. For example, the concept "Collection Variables" has an *Abstraction* link to the node "Trace Species". When the k-net determines if the node "Trace Species" should be offered, value links emanating from "Trace Species" are relevance processed. Upon locating the first value link from "Trace Species" considered relevant, the link from "Collection Variables" to "Trace Species" is then declared relevant. Relevance processing is recursive and if any links emanating from "Trace Species" are also *Abstraction* links, each destination node of each Abstraction link is processed in turn until a relevant link is located. As stated above, the node "Trace Species" is a valid value selection for the *ISA-S* node "Collection Variables". However, it is also an abstraction of more specific value concepts. This type of node, one which represents a value but is not a leaf value node, is referred to as an *ISA-V* node. As with *ISA-S* nodes, the *ISA-V* node began as a *Value* node but was modified when a request was received to define an *ISA* link with the *Value* node as the destination.

Leaf values of *value trees* are generally *Value* nodes, although there are exceptions. *Value* nodes represent a valid value for the attribute node that is the root of the tree. They represent the most specific domain concept known for any attribute and, hence, do not have links to more specific concepts. The only exception to this rule is when value range type information is known. In such a case a *Value-Selection* link is defined for the node pointing to an appropriate *Structure* node. The *Structure* node then has *Fact* links defined that point to *Fact* nodes representing specific ranges of values for the associated *Structure* node. Upon selection of the *Structure* node, a routine is invoked that displays relevant value range information, allowing the user to indicate value ranges of interest. In this way, the *value tree* structure allows multiple levels of data abstraction with different *metadata* items having varying levels of attribute value abstraction.

Although the node value structure has been described in terms of a tree (which it appears to be upon first glance), there is nothing to prevent links from entering the structure at any location from outside, or to prevent links within the value tree from pointing outside the structure. The latter condition is almost always the case although these types of links are generally not involved with the selection of a value for an attribute, but are links to other related concepts. It is important to realize that the *value tree* structure allows setting attributes to desired values, with only relevant and valid values being offered to the user. Thus, the user browses relevant data while forming the query.

As stated above, the k-net may be thought of as a series of *value trees* grouped together through the use of *Structure* nodes and *Structure* links and representing category headings or other structural components. While defining a *Value* link causes a companion *ISA* link to be defined in the reverse direction of the *Value* link, not all links cause the creation of companion links. However, certain analysis procedures require traversing structural and value relationships within the k-net. Therefore, to provide this capability, a *Reverse* link must be defined in the opposite direction of the *Structure* link. It is important to understand that *Reverse* links are not used to model relationships in the k-net and are not used in any way to control the navigation process; they are only used to provide for required analysis capabilities.

5.2 SYNTACTIC REPRESENTATION

While assigning values to attributes is a large part of the query formulation process, it is also necessary for these bindings to be grouped and joined in a logical manner. *Paren* and *Operator* nodes are used to implement this capability for English-like syntax representation in the k-net, and these nodes have inherent capabilities that impact query syntax. For example, a right paren cannot be offered unless there is a corresponding open left paren, or a conjunction cannot be offered until

an attribute/value binding is completed.

5.3 NON-QUERY INFORMATION REPRESENTATION

Information nodes and links provide the capability to provide non-query related information about nodes to which the user has navigated. Information nodes are usually entry points into the tutorial system and provide such information as drawings, explanations, and technical specifications. In the IIMS prototype, this type of information is provided by hypermedia presentation so the user can use the entry point as the beginning location for an in-depth exploration of the given subject, or related subjects. Selection of these nodes does not directly add anything to the query.

5.4 ALTERNATIVE NAME REPRESENTATION

Two links are used within the k-net to represent alternative names for nodes: *Acronym* and *Alias*. The presence of an *Acronym* link indicates that a certain node has an associated acronym, and, since these links can have relevances associated with them, a single node may have more than one acronym. In the IIMS prototype, a user can request that a node be displayed using either its full name or its acronym. If acronym is selected, the IIMS prototype processes relevance instances associated with each *Acronym* link until one is found to be relevant. The acronym name given to the destination node will then be used as the display name of the node.

The *Alias* link is used to allow alternative names for nodes. Such a situation is made necessary given the condition where one node represents a single concept but different domains may have different names for that concept. Obviously, this link is used only when the concepts are exactly the same. If concepts are slightly different, a different node will have to be defined.

5.5 LINK RELEVANCE PROCESSING

Within the k-net, a single concept (i.e., node) appears only once within the semantic network. However, during the navigation process, it may seem to the user that many nodes exist expressing the same concept. Such a perception is caused by choices the user is offered upon selection of a concept. Remember that choices offered are dependant upon the path the user has taken to arrive at the concept. As briefly discussed above, this is achieved through the use of relevance descriptions associated with each link. Therefore, when a user arrives at a node, the k-net processes each link emanating from the node. Only those links determined to be relevant are used to define the next choices offered to the user. For display purposes, this conditional relevance technique is used to reduce the number of nodes and links typically associated in a highly interconnected network.

Associated with each link is a list of RELEVANCE conditions. When a link creation request is instantiated, the system determines if a similar link-type already exists (i.e., the same as that requested other than the associated RELEVANCE condition). If such a link is found, a new link is not created. Instead, the RELEVANCE condition of the new link request is added to the existing link. As a result, unbridled proliferation of links is controlled. Currently, two types of RELEVANCE conditions are used: COMPLIANCE and NON-DISPUTING.

If a RELEVANCE condition is specified as COMPLIANCE and the condition fails to be satisfied, the link is rejected without further processing. However, if a RELEVANCE condition is NON-DISPUTING, then as long as the link's relevance is not disputed by the current state of the query (i.e., a variable set by the query process does not have a value conflicting with that specified in the RELEVANCE condition), then failed relevance does not cause the processing of the link to be terminated. Such a process is made necessary to allow specification of RELEVANCE conditions that will not cause rejection of the link if variables used in the specification of the RELEVANCE condition have not been set by the query process. For example, if a RELEVANCE condition is specified by the fact that the program must be NIMBUS and a NON-DISPUTING condition is present, then the link's relevance is rejected only if the program is specified to be something other than NIMBUS. If the program variable has not already been set, link relevance is not rejected. However, if the RELEVANCE condition is COMPLIANCE, then the link's relevance is accepted only if the "Programs" variable is set to NIMBUS.

Determination of link relevance also involves processing the non-reentrant state (see section 5.5.1) of the link's destination node and then processing RELEVANCE conditions associated with it. If the link passes the non-reentrant filter test, then RELEVANCE conditions associated with the link are processed in two steps: *view filtering* and *relevance predicate processing*. Results of this two-step process can be used to immediately accept or reject the link or indicate that no information concerning link relevancy can be provided. In the second case, the next RELEVANCE condition is processed. This process continues until the link has either been accepted or rejected, or there are no more RELEVANCE conditions to process; in such a case the link is accepted as relevant. The second process step is more involved and takes more time than the first. In this way, *view filtering* and *non-reentrant processing* act as rapid filters that eliminate many links from the more intensive and time consuming *relevance predicate processing*. Each of these processes are discussed more

fully in the following sections.

5.5.1 Non-Reentrant Nodes

It is necessary that the k-net be able to represent the fact that under certain conditions some nodes cannot be reentered during the navigation process. In the current k-net, most nodes representing variable binding may not be reentered under an AND condition. This requirement can be illustrated when one considers the possibility of a single data set having two Analysis Product Qualities: A AND B; obviously, such a state is untenable. However, it is possible for a data set to have an Analysis Product Quality of A OR B. Therefore, the node "Analysis Product Quality" must have non-reentrant capability so that it will not again be offered to the user if it has already been offered within the scope of the current AND condition, but may again be offered under other conditions. Non-reentrant capabilities have been implemented in the current k-net, and checking this condition is the first step in link relevance processing, needing to be performed only once per link. If the destination node of the processed link indicates that the non-reentrant attribute is activated, syntax of the current state of the query formed through navigation is analyzed to determine if a restrictive condition exists. If so, the link is rejected and processing on this link ceases. If the link passes the *non-reentrant processing* test, individual RELEVANCE conditions are processed until determination can be made about the relevance of the link. The first step in this process is passing the RELEVANCE condition through a *view filter* and then, if successful, relevance processing concludes with *relevance predicate processing.*

5.5.2 View Filtering

A view in the IIMS prototype is a set of nodes and links which may be composed of, but are not limited to, nodes and links of the system-wide knowledge structure. Each view is the encapsulation of knowledge relevant to a particular interest or domain and contains domain relevant concepts, relationships, and terminology not included in the system-wide knowledge structure. User views, then, represent sets of base knowledge structure changes required to reflect user-specific domain knowledge. Views provide the user with the capability to form queries in a familiar environment and exclude information irrelevant to interests defined by the view. In the IIMS prototype, views are used to aid in the efficient presentation of relevant information to the user, and to present the information in an appropriate form for the specified user's interests. Separate views defined in the IIMS prototype form a hierarchical inheritance structure in which a view inherits all associated subview modifications. For example, a Physics view might be composed of all concept and relationships defined in the Astrophysics, Geophysics, and Atmospheric Physics views since the generalized Physics concept consists of all these specific domains.

Each RELEVANCE condition contains a list of views for which the associated link is considered relevant. To determine the relevance of the current RELEVANCE condition, this view list must be compared against the view in which the k-net is currently operating. A link passes the *view relevance filter* if one of the views or subviews for which the link is relevant is currently active. To use the example above, if the current view of the k-net was Physics and one of the views of the current RELEVANCE condition is Astrophysics, the RELEVANCE condition would pass the *view relevance filter*. Of course, the converse would not be true. Once a view passes the *view relevance filter*, view filtering of the link can be terminated and relevance processing can continue into the next phase. If no views contained in the RELEVANCE condition are found to be relevant, then processing of the RELEVANCE condition is terminated.

5.5.3 Relevance Processing

Once the link has passed both *view* and *non-reentrant filtering*, it enters the next phase of relevance processing. As discussed above, there are two types of relevance instances: COMPLIANCE and NON-DISPUTING. The RELEVANCE conditions associated with the link are grouped according to these types. All COMPLIANCE relevance conditions must be processed and all must be satisfied. If one of these relevance instances fails, relevance processing of the link ceases and the link is rejected. If all COMPLIANCE tests are completed successfully, NON-DISPUTING relevance conditions are then processed. In this instance, a negative result does not cause the rejection of the link while a positive response results in the link being immediately accepted.

Associated with every RELEVANCE condition is an EVALuable predicate that determines conditions under which the link is relevant and a set of variables that must be set before the predicate is EVALed. Values to which variables will be set are determined from the current state of the k-net. While navigating the k-net, certain attribute/value bindings and relationships between them have been formed. For example, when a user navigates to the "Programs" node, all relevant programs are offered for selection. Once a specific program has been selected, a binding for the node "Programs" has been made. Once value bindings of specified variables have been set based on the current state of the k-net, the relevance predicate is EVALed. The resulting action depends on the type of RELEVANCE condition being processed. It is possible for an attribute to have more than a single value set during the navigation process

(through the use of an OR operator) and thus, if the result of the EVALuation of the relevance predicate is NIL, then, maintaining the original semantics of the query, alternative values for the variables are tried and the predicate is reEVALed. This process continues until the predicate returns T or there are no more alternative values for the variables.

Use of the EVALuable predicate introduces an element that is less flexible than desired. Originally, relevance was determined based strictly on the attribute/value bindings established during the query process. An elaborate process that involved "covering" all facts associated with link relevance was used to determine if the current navigation path supported acceptance of the link as relevant to the current state. This technique worked well in most cases since query formation is, basically, the process of indicating desired values of certain attributes describing information to be retrieved. However, this method proved less robust than required for more involved selection criteria such as conditions under which links to the *Paren* nodes are valid. This case involves more than simple attribute/value bindings but rather, involves qualifications of a semantic nature. Therefore, to provide the capability to handle these more complex relationships, the EVALuable predicate expression was employed, making link relevance more difficult to set or change. Yet, with certain tools that we will provide, the problem should be manageable.

This limitation most frequently arises when the user attempts to define his own link. To do this users must specify the conditions under which the link is relevant. By providing the capability to allow the user to input attribute/value bindings and associated conjunctions through the use of a form system, most of the difficulty can be eliminated. This utility will then convert user input into an appropriate predicate to be included with the link. However, for those instances when this procedure will not suffice, the user will be able to directly input the code satisfying the link's relevance requirements. This issue will be studied further, but in any case, original source code will not have to be changed nor will the system have to be rebuilt if RELEVANCE conditions of a node or group of nodes need to be altered. Hence, original requirements placed on the knowledge structure are satisfied.

6.0 AUTOMATED KNOWLEDGE STRUCTURE UPDATES

This section focuses on the primary requirement that future information management systems be able to incorporate new data and information in an automated fashion. This requirement is particularly important when one considers the projected rates at which data will be captured, the projected volumes of data that will be stored, and the secondary requirement[1] that all updates be managed in a completely dynamic environment; that is, the system must be able to accept continuous updates at the same time multiple users are accessing the system. As we have suggested earlier (Carnahan, Corey, & Snow, 1989), several alternatives for implementing automated updating are possible.[2] Following sections describe in detail the initial approach we have taken.

6.1 AUTOMATED UPDATE REQUIREMENTS

As we have already suggested, the requirement to automatically update data, as well as information about the data, will drive the design and implementation of advanced information management systems. The approach we have taken to the issue of automated data/information updates is based on the premise that any future system will have to consider not only which, if any, existing data parameters to classify as *metadata*, but also how those parameters may be translated into more meaningful information related to other, existing information.[3] Obviously, the second objective, to relate information, is of paramount importance if the user is to have any success at all in accessing the large volumes of available data. Figure 4 shows our functional approach to IIMS implementation. Those components we believe are best suited to the application of intelligent systems are indicated. For purposes of the discussion at hand, we will focus on the three components comprising the *Information Update System.*

In general, we view the process of updating information about data to be two-phased. The first phase includes processing of the update message by the *Knowledge Encapsulation System (KES).* Described in more detail in section 6.2, KES functionality includes the capability to infer *metadata* information from data parameters provided by the update message, and then relate that information to already existing information resident in the *Knowledge Net.* The *Update Processor* is not only responsible for providing the update message to the KES, but, just as important, it is responsible for translating the output of the KES into something the *Knowledge Net* can understand Neither process, information inference nor translation, is trivial.[4] As discussed

[1]Some may argue its primacy.

[2]The key criterion here is that the user's search efficiency and effectiveness are minimally impacted by the the ability of the update system to provide adequate *metadata* and to establish appropriate links to already existing information.

[3]While the issue of updating the database where the actual data are stored must also be addressed, it is the not the focus of our current work.

[4]It is important to realize, however, that we view the two

earlier, the *Knowledge Net*, or *metadata* base, is that portion of the IIMS where *metadata* information is stored. When users access the IIMS and browse or query the system, they deal almost exclusively with the *Knowledge Net*. The *Knowledge Net* includes procedural information concerning navigation paths and

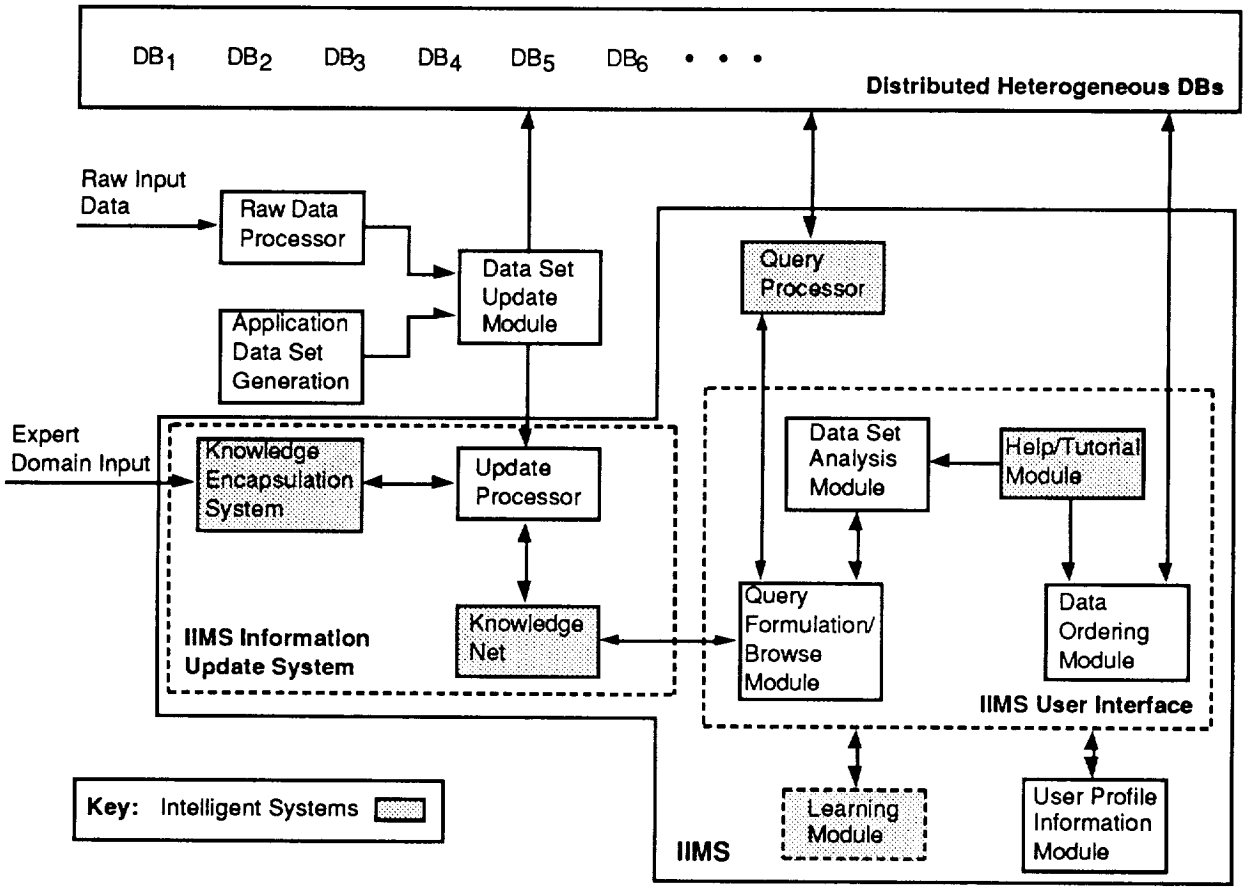

Figure 4 IIMS Implementation Approach

associated dynamic relevance weights used to present users with the most appropriate information given the user's interests and present location within the *Knowledge Net*. The *Knowledge Net* also contains control knowledge necessary to form the appropriate node and link updates once new information has been inferred and then transferred by the *Update Processor*.

6.2 KNOWLEDGE INFERENCE FROM METADATA

Update messages are generated whenever new data sets are input and they serve to inform the system of the various known data set parameters.[5] The vast majority of these new data sets will not add any new knowledge concepts or relationships to the knowledge structure since information known about the data set is an abstraction of the real data. In most cases, this abstract information will not actually include information about data set range values and hence, each data set collected by an instrument during a certain mission phase will only differ from a previous data set by temporal and spatial values.

Automated processing of update messages proceeds in two phases. The first phase (*analysis phase*), involves extracting a set of knowledge concepts and relationships represented by the data set parameters existing in the update message. It should be noted that this process involves more than a simple, direct extraction of attribute/value pairs; it also involves reasoning about the data parameters specified in the update message. The second phase (*distribution phase*), takes this reasoned set of knowledge concepts and relationships and integrates it into the basic knowledge structure, as

processes as independent.

[5] As Figure 4 indicates, such data sets could be raw data that has not been processed to any higher level than level 0 or higher level processed data input from data users. In the latter case, we would expect that some level of *metadata* other than basic data set parameters would already exist.

well as into other knowledge used by the system. The *distribution phase* is also responsible for storing data reflecting knowledge structure changes so that user views will be updated appropriately. These two phases are examined in more detail in the following sections.

6.2.1 Inferring Metadata - The Analysis Phase

Analysis of the update message, the first phase of automated update processing, must extract all relevant concept and relationship information contained, either explicitly or implicitly, in the update message. To do this requires the ability to incorporate expert information about relevant scientific domains into the system for use in reasoning about appropriate *metadata* and *metadata* relationships.[6] Information such as data set interrelationships, the relationship of terminology used by different domains, and information concerning the potential relevancy of new concepts and relationships (including the view(s) under which the relationship is valid) must be encapsulated in a form that can be applied to the update message and that can be altered when necessary. An expert system[7] is the most likely choice for this type of knowledge representation as long as the expert system development environment allows for easy modification of the knowledge it contains and can be invoked from a program.

The process of updating the expert system does not have the same stringent restrictions as those placed on updating the knowledge structure since the expert system will only be used to process update messages. Accessing of the knowledge structure will not be impacted if the knowledge encapsulation process needs to be shut down for a period of time to effect modifications to the expert system. The real problem then becomes the ease with which the expert system can be updated and how such updates are distributed.[8]

It is important to note that the set of knowledge concepts and relationships generated by this phase is independent of the knowledge representation scheme used in the actual knowledge structure; if the knowledge structure representation system changes, the module performing the *analysis phase* will not have to be altered. Additionally, the translation of information generated during the *analysis phase* to a form that can be input to the knowledge structure is totally dependent on the knowledge structure implementation. An example of the types of information we expect to be provided from the KES is given in section 6.2.3.

6.2.2 Reasoning and Knowledge Structure Modification - The Distribution Phase

The *distribution phase*, the second phase of automated update processing, uses the set of knowledge concepts and relationships generated by the *analysis phase* to make appropriate modifications in overall system knowledge. This process involves updating (1) the knowledge structure accessed by users, (2) system and user views used in navigation, and (3) other knowledge used by the system.[9]

Five types of knowledge structure updates can be generated from modifications to the expert system or from the *analysis phase*. They include:

1. Adding new relevance conditions to already existing relationships;
2. Adding new relationships;
3. Adding new knowledge concepts;
4. Deleting existing relationships;
5. Deleting existing knowledge concepts.

The first three update types can be generated from either the *analysis phase* or by a modification of the expert system; the last two can only be generated from a modification to the expert system. As stated above, these updates can affect three knowledge areas in the system, the base navigation knowledge structure, the system and user views, and other knowledge structures. The first two will be more fully developed below while the third area has yet to be defined in the current system.

Modification of the base knowledge structure is straightforward and is handled no differently than the normal process of creating knowledge concepts and relationships handled by the knowledge representation scheme. The set of knowledge concepts, appropriate relationship information, and the conditions under which both concepts and relationships are relevant are generated during the *analysis phase*. All that must be done is translate

[6]Obviously, selection of what scientific domains are relevant is somewhat subjective and influenced largely by user interest.

[7]Or more globally, a reasoning system.

[8]One could speculate wildly here concerning the potential applicability of automated knowledge system updating or learning systems. At this time, we believe such updates are likely to be generated manually, at least initially.

[9]Other knowledge might include that used by a learning system to interpret user actions in terms of goals. We can envision, for example, a parallel system that monitors user activity to determine abstract types of information the user may be interested in. Such information could then be incorporated into the user's profile and views to make future sessions more efficient.

this information into the appropriate requests to the knowledge system. Knowledge structure modification requests take the same form as those initially used to create the original knowledge structure; indeed, the same methodology to create new nodes and links is used. Since an initial requirement of the knowledge system was that it must have the ability to be dynamically updated, execution of knowledge structure modification requests can be accomplished with no operations interruption to current users of the system.

The more complex task resulting from updates is the modification of user views. User views are the overlying representation of a set of additional or different knowledge concepts and relationships than those defined by the base knowledge structure. Therefore, user views reflect knowledge concept and relationship change sets applied to the base knowledge structure, creating access environments desired by individual users. When a modification occurs to the base knowledge structure, the change may be outside the domain or set of domains represented by any user view.[10] In such a case, no changes in user views need be reflected. However, it is highly possible that a base knowledge structure modification would affect the knowledge concepts and relationships within any particular user view. In this case, several situations may be encountered. For example, users may not agree with the change or the change request represents an inadequate or inaccurate relationship within a user's domain of interest and hence, it would not be appropriate to include the change in that user's view. However, in most cases, the user will likely want changes to the base knowledge structure reflected in his or her view so that the most recent information is available. If such is the case, nothing special needs to occur. In the first situation, several modifications would be required.

In addition to user view updates, it must be remembered that existence of a significant potential for a large number of user defined views could result in an inordinate amount of time being required to update all user views concurrently. Obviously, such a condition is undesirable. The following methodology is used to address this problem. When an update request is received by the base knowledge structure, the update is logged in a database keyed, for example, on time of request initiation and possibly the potential user view(s) involved. As a result, when a user selects a view, the update database is checked to determine which updates have been executed since the last time the view was instantiated and which of those are applicable within the scope of the view. (Remember, as suggested earlier, most update messages will not involve actual changes to knowledge concepts or relationships defined in the base knowledge structure and hence, the number of updates potentially relevant to the user should be small. Changes reflecting temporal and spatial differences in data sets will not invoke an update resulting in a potential user view modification.)

If modifications affecting the user's view have been logged, the user is notified that potentially relevant changes have occurred. The user then has the option of addressing the changes at the current time or processing them at a later time. When the user decides to address the changes, a list of request changes are presented and the user can select those considered relevant within the current view. When the selection has been made, appropriate changes will be made to the view. If the change is accepted, nothing will have to be done to the user's view since the change has already occurred in the base knowledge structure. However, for those changes that are rejected, appropriate modifications need to be made in the user's view to undo modifications already made to the base knowledge structure. If the user chooses to postpone addressing the changes, the user's view will have to be modified to undo the changes which were made to the base knowledge structure.

Since most users will likely want to accept changes automatically, a facility is provided to allow the user to select a mode of system operation in which changes are immediately instantiated within the appropriate user views. Or, if desired, the user can be prompted for each change before incorporation.

6.2.3 Automated Update Message Processing - An Example

To help clarify the process of automated update message processing and potential difficulties that may be encountered when implementing such a capability, several example update messages are provided.[11] Additionally, further clarification of reasoning required for automated knowledge structure updating is given in terms of potential *analysis phase* output.

The first example update message is taken from an amateur observation of a comet under the jurisdiction of the International Halley Watch organization. Most of the

[10] Such modifications might occur, for example, when new missions are launched and the base knowledge structure is modified to reflect the new information.

[11] Example update messages represent information extracted from the headers of various data files included in the *NASA, Space Science Sampler, Volume 2: PDS Interactive Data Interchange CD-ROM*. These data files represent various data and header formats in common usage in scientific domains. We are assuming that types of information contained in the headers represent the types of information to be included in update messages for processed data sets.

attribute names are extracted directly from the file.

Update Message 0001

```
Object=P/CROMMELIN
File-Num=800200
Date-Obs=29/12/83
Time-Obs=.7600
Date-Rel=13/12/85
DiscipIn=Amateur
Long-Obs=999/99/99
Lat-Obs=+99/99/99
System=85000000
Spec-Evt=F
Dat-Type=Visual Mag. Est
Instrume=Newtonian
Aperture=.26
Fratio=6.
Power=63
Origin=Jet Propulsion Lab
Comment=Observing Site Unknown
Associated-File=AMATE001.FIT
Origin=JPL
```

To correctly process this update message, processing occurring during the *analysis phase* must be able to determine what type of observation was made and the identity of the observation target. The first piece of information is supplied by the attribute *Object* which identifies the observation target as P/CROMMELIN. Obviously, to correctly incorporate the object into the knowledge structure, the type of object which P/CROMMELIN represents must be determined. For example, it must first be determined that P/CROMMELIN is a comet, and that as a comet, it has certain other attributes (e.g., orbit, size, next predicted encounter, previous encounters, components).

The observation type is provided by the attribute *Dat-Type* and is identified as Visual Mag. Est. For the uninitiated user to make sense of this, the abbreviation must be expanded and its meaning understood. For example, because the observation involves measuring magnitude, the system should be able to reason what types of instruments might be used to make this type of observation. Such a capability aids additional processing of the message and aids the appropriate placement of the observation within the knowledge structure.

The *Instrume* attribute indicates that the measuring instrument is a Newtonian. The system will have to reason that a Newtonian instrument, which can be used to perform magnitude measurements, is a telescope, and that, as a telescope, it has certain characteristics. Within this context, processing of the update message can continue. Therefore, the attribute *Power*, when encountered, is now interpreted within the appropriate domain context (i.e., telescope), suggesting that *Power* indicates magnification used in the observation instead of an electrical power setting or other possible interpretation. Another example of inferring missing information from context can be seen when examining the attribute *File-Num*. This attribute provides a unique identifier but only within the organization responsible for the data. Meta-information inferred from *File-Num* must be determined from domain information contained within the KES and using various clues resident in the update message. From these clues it can be determined that *File-Num* refers to files maintained by the International Halley Watch.

However, this type of reasoning is not the only type required during the *analysis phase*. Different data formats use different attribute names to represent the same information or the same name to denote different information. Additionally, attribute formats indicating spatial and temporal information do vary, and the system must be able to understand how to interpret each type of information. This situation can be illustrated when comparing the update message 0001 with update message 0002.

Update Message 0002

```
Spacecraft_Name=Voyager_1
Mission_Phase=Jupiter_Encounter
Target_Body=Jupiter
Frame_Id=1309J1-059
Spacecraft_Clock_count=14641.14
Spacecraft_Event_Time=1979/01/06-
05:32:34
Earth_Received_Time=1979/01/07-00:20:51
Instrument_Name=Narrow_Angle_Camera
Instrument_Scan_Rate=1:1
Instrument_Shutter_Mode=NAONLY
Instrument_Gain_State=Low
Instrument_Edit_Mode=1:1
Instrument_Filter_Name=Orange
Instrument_Filter_Number=3
Instrument_Exposure_Duration=0.96000
Associated-File=C1464114.IMG
```

Notice the different names and formats for observation time, instrument used, and spatial specifications. In addition, notice how the attribute *Frame-Id* in update message 0003 uses a different format than update message 0002.

Update Message 0003

```
Spacecraft_Name=Voyager_1
Mission_Phase=Jupiter_Encounter
```

Target_Name=Jupiter_Magnetosphere
Frame_Id=16269.49
Frame_Period=48 <seconds>
Spacecraft_Clock_count=16269.49
Spacecraft_Event_Time=1979/060-12:24:36
Instrument_Name=Plasma_Wave_Spectrometer
Instrument_Mode=Waveform_Receiver
Instrument_Sampling_Rate=28800
Instrument_Lost_Samples=128
Associated-File=C1626949.IMG

It is possible that information used in the reasoning process during the *analysis phase* is inadequate to understand all pieces of the update message. When an update message is received in which some attribute or attribute value is unknown, further aid in interpreting the information will be required. In this case, the update message would likely be placed in a temporary buffer and a clarification request sent to the system operator. The update message is processed when the system operator responds to the clarification request. The operator's response might be as simple as identifying the unknown attribute in terms of an ISA-relationship with a known knowledge concept. For example, if reasoning during the *analysis phase* could not identify P/CROMMELIN, the system operator could inform the system that it is a comet and the update message could then be processed.[12] However, it is possible that a more complex alteration to the knowledge used by the reasoning system is needed. As a result, the expert system would have to be modified to reflect the new knowledge concept and its relationships to existing metainformation resulting in updates being processed as describe earlier.[13]

To better understand the nature of *analysis phase* processing and its relationship to automated updating of the knowledge structure, we have begun to closely examine what possible structure output of the *analysis phase* may take. For purposes of this discussion and to help the conceptual framework of analysis phase processing, initial results of our examination are presented here in terms of one of the example update messages provided earlier. Update message 0001 is reproduced below.

Update Message 0001

Object=P/CROMMELIN
File-Num=800200
Date-Obs=29/12/83
Time-Obs=.7600
Date-Rel=13/12/85
DiscipIn=Amateur
Long-Obs=999/99/99
Lat-Obs=+99/99/99
System=85000000
Spec-Evt=F
Dat-Type=Visual Mag. Est
Instrume=Newtonian
Aperture=.26
Fratio=6.
Power=63
Origin=Jet Propulsion Lab
Comment=Observing Site Unknown
Associated-File=AMATE001.FIT
Origin=JPL

The first step of the analysis process would examine the *Object* attribute, the first attribute provided in the update message, and would relate the attribute to the concept *target body*.[14] Establishing this relationship determines where in the knowledge structure *target body* is located. In this example, we assume that *target body* is directly related to the concept *Science Interests*.[15] Knowing this information results in the generation of the first segment of relevant concept/relationship information. Part of this information is given below:

Concept 1: Science Interests
Concept 1 Type: Structure
Concept 2: Target Body
Concept 2 Type: Structure
Relationship: Structure
Relevance View: Cometary Studies

This information identifies applicable concepts, their types, relationships existing among them, and associated relevance conditions. At this point in the process, only the appropriate relevance view is known. After all concept/relationship information is known, a comprehensive relevance condition would also be generated.

After the attribute has been processed, the attribute's value is processed. The system would then reason about P/CROMMELIN taking into account that it is a type of *target body*. The fact that P/CROMMELIN is a *comet* can then be determined.[16] Using the concept

[12]Such a scenario assumes the system operator is either knowledgeable concerning basic domain information or has access to reference material describing such information.

[13]This scenario might require contact with a domain expert to help define appropriate domain knowledge.

[14]The concept *target body* represents one node in the existing knowledge structure. In reality, the system would attempt to relate the attribute to as many concepts as possible.

[15]Keep in mind that the term 'concept' is used here to refer to any node in the knowledge structure. The term 'relationship' refers to nodal links.

[16]The actual relationship would be *ISA*.

target body as a starting point, the concept *comet* can then be located within the knowledge structure. In this case, *comet* has a direct relationship with *target body* and hence, the following concept/relationship information is generated.

Concept 1: Target Body
Concept 1 Type: Structure
Concept 2: Comet
Concept 2 Type: Value
Relationship: Value
Relevance View: Cometary Studies

Note that the relationship is now of the type *value* since *comet* is a valid value for the concept *target body.*

The next piece of information follows directly, linking the concepts *comet* and *P/CROMMELIN.*

Concept 1: Comet
Concept 1 Type: Value
Concept 2: P/CROMMELIN
Concept 2 Type: Value
Relationship: Value
Relevance View: Cometary Studies

Given this new concept/relationship structure, a user could indicate that the desired value for the concept *target body* could be either *comet* (all data sets pertaining to any comet are selected) or *P/CROMMELIN* (only those data sets dealing with this specific comet are selected). The user is allowed to specify the more general concept *comet* for the value of *target body* and then, if desired, return at a later time and define the concept more specifically by selecting a specific comet or group of comets of interest.

As with the concept *target body*, most concepts can only be assigned a single value.[17] However, there are concepts for which multiple values are acceptable. One of these concepts is *Instrume.* It is possible that some data sets can be generated by a collection of instruments working together.[18] Therefore, the concept *Instrume* would have to represented with multiple value capability in the output of the reasoning system .

In our example, when analysis processing arrives at the concept *Instrume*, it must be reasoned that this concept is the same as the concept *instrument* to be found in the knowledge structure. Like *target body, instrument* is related to the concept *Science Interests.* The following information would then be generated.

Concept 1: Science Interests
Concept 1 Type: Structure
Concept 2: Instrument
Concept 2 Type: Multiple Value Structure
Relationship: Structure
Relevance View: Cometary Studies

As before, the value of the *Instrume* attribute is then processed. The system determines that *Newtonian* is a type of the concept *telescope* which is a type of *instrument.* These concepts are located in the knowledge structure and the following information is generated.

Concept 1: Instrument
Concept 1 Type: Multiple Value Structure
Concept 2: Telescope
Concept 2 Type: Value
Relationship: Value
Relevance View: Cometary Studies

Concept 1: Telescope
Concept 1 Type: Value
Concept 2: Telescope Type
Concept 2 Type: Structure
Relationship: Qualification
Relevance View: Cometary Studies

Concept 1: Telescope Type
Concept 1 Type: Structure
Concept 2: Newtonian
Concept 2 Type: Value
Relationship: Value
Relevance View: Cometary Studies

Finally, there are times when concept defining information will need to be generated. Using update message 0001, one example is to define value units for the attribute *aperture. Aperture* must be processed in the context that it represents a qualification of the specified *instrument.* The reasoning system indicates that *aperture* specifies the aperture size of the *instrument telescope.* In addition, the reasoning system determines that defining information is required. As a result, the following information would be generated.

Concept 1: Telescope
Concept 1 Type: Value
Concept 2: Telescope Aperture
Concept 2 Type: Structure
Relationship: Qualification
Relevance View: Cometary Studies

Concept 1: Telescope Aperture
Concept 1 Type: Structure

[17]Within an AND condition, alternative values can be selected using an OR condition.

[18]A photometer attached to a telescope is one example.

Concept 2: .26
Concept 2 Type: Value
Relationship: value
Relevance View: Cometary Studies

Concept 1: Telescope Aperture
Concept 1 Type: Structure
Concept 2: Telescope Aperture Information
Concept 2 Type: Information
Relationship: information
Information: The size in meters of the effective aperture of the specified telescope.

When the user navigates to the value .26 for *aperture*, an *information* concept is provided which defines the value.

In the course of generating this type of output for each update message, it becomes clear that many redundant specifications exist. Such a situation does not pose a problem since the knowledge structure checks specified relationships and concepts, and if an exact match is found, the concept or relationship will not be redefined. It is important for two reasons that the complete set of concepts and relationships included in the update message be generated each time: the *analysis phase* does not maintain information about which relationships have already been defined, and users may not have the specified relationship within their views.

When all concept/relationship information is generated, the reasoning system then knows under what conditions each link is relevant. In this case, the following information has been understood:

Target Body - Comet or P/CROMMELIN
Instrument Type - Newtonian
Instrument Aperture - .26

This information would then have to be combined with other information to form the appropriate relevance condition for the link. As indicated earlier, a relevance condition can be more than a simple specification of attribute/value bindings. It can also contain complex logic based on other factors (see section 5.5).[19]

7.0 SUMMARY AND CONCLUSIONS

We have attempted in this paper to provided the underlying framework for what we consider to be two of the most crucial problems facing future, advanced information management systems: the design and implementation of an efficient knowledge structure supporting sophisticated user information access and manipulation, and the incorporation of automated knowledge structure update processing to provide an effective means for accomplishing knowledge structure evolution. As a result, the underlying knowledge structure for an earlier IIMS prototype has been extended and modified to handle not only semantic but also syntactic information, a capability made necessary by the requirement for the IIMS to be updated automatically. Our approach to knowledge structure development and modification has resulted in a significantly more flexible and easily modified knowledge structure that will be used in the future to continue to examine issues related to automated data and *metadata* updates for very large data/information systems.

[19]Relevance condition specification is beyond the scope of this paper.

REFERENCES

Carnahan, R.S., & Corey, S.M. (1989). Advanced information management and global decision making. *Proceedings of the Conference on Earth Observations and Global Change Decision Making: A National Partnership*, in press.

Carnahan, R. S., Corey, S. M., & Snow, J. B. (1989). A rapid prototyping/artificial intelligence approach to space station-era information management and access. *Telematics and Informatics, 6(3-4)*, 273-297.